R code for Data Science for Beginners

Day 3: Individual Exercise

AUTHOR
Graham Jones

PUBLISHED
September 20, 2025

1. Vectors

Create an object called `vec.a` which is a vector consisting of the numbers, 1, 3, 5, 7. You need to use the c function.

```
vec.a <- c(1, 3, 5, 7)
```

Create a vector called `vec.b` consisting of the numbers, 2, 4, 6, 8.

```
vec.b <- c(2, 4, 6, 8)
```

Subtract vec.b from vec.a

```
vec.a - vec.b
```

```
[1] -1 -1 -1 -1
```

Create a new vector called vec.c by multiplying vec.a by vector vec.b

```
vec.c <- vec.a * vec.b
```

Create a new vector called vec.d by taking the square root of each member of vec.c

```
vec.d <- sqrt(vec.c)
```

What is the third element of the `vec.d` vector? Find out using square bracket. Note that since this is a vector, you only need to provide a single number inside the brackets.

```
vec.d[3]
```

```
[1] 5.477226
```

Create a new vector called vec.e consisting of all the integers from 1 through 100. You should use the seq function, rather than writing down all the 100 integers individually.

```
vec.e <- seq(1,100)
```

The mean function calculates the arithmetic mean of the numbers stored in an object. Using the mean function, calculate the mean of the `vec.e` vector.

```
mean(vec.e)
```

```
[1] 50.5
```

As we saw in the joint exercise, the sum function calculates the sum of all the elements in an object. Calculate the sum of the `vec.e` vector.

```
sum(vec.e)
```

```
[1] 5050
```

The length function returns the number of elements stored in an object. Using the length function, find the number of elements stored in the `vec.e` vector.

```
length(vec.e)
```

```
[1] 100
```

The mean of an object can be obtained by sum(X)/length(X) because the definition of the mean is the sum of elements divided by the number of elements. Now, using the sum and length functions, calculate the mean of the `vec.e` vector. Compare the answer with that obtained with the mean function

```
sum(vec.e) / length(vec.e)
```

```
[1] 50.5
```

We have learned that the by argument specifies an increment. For example,

```
seq(from = 0, to = 10, by = 2)
```

```
[1]  0  2  4  6  8 10
```

This creates a sequence that starts from 0 and ends with 10, and with an increment of 2.

Now, create a new object called olympic which is a sequence that starts from 1896 and ends with 2012, with an increment of 4.

```
olympic <- seq(from = 1896, to = 2012, by = 4)
```

How many elements does the olympic vector contain? That is, what is the length of this vector? Find out by applying a function (not by manually counting the number of elements).

```
length(olympic)
```

```
[1] 30
```

So there are 30 elements in the olympic vector. Display all the elements contained in the olympic vector. These are the years where olympic games were (supposed to be) held. Display the contents of the olympic vector.

```
olympic
```

```
 [1] 1896 1900 1904 1908 1912 1916 1920 1924 1928 1932 1936 1940 1944 1948 1952
[16] 1956 1960 1964 1968 1972 1976 1980 1984 1988 1992 1996 2000 2004 2008 2012
```

Find out how many olympic games will have been held by the year 2400. Use the length and seq functions.

```
length(seq(from = 1896, to =2400, by = 4))
```

```
[1] 127
```

2. Matrices

Create a new vector called `v1` consisting of the following numbers: 1, 3, 5, 7, 9, 11

```
v1 <- c(1, 3, 5, 7, 9, 11)
```

Find out the length of this vector (Don't count the numbers by hand; use an appropriate function).

```
length(v1)
```

```
[1] 6
```

We will convert this vector into a matrix. That is, we will rearrange this vector so that it will have two dimensions (rows and columns). Since this

vector has 6 numbers, if we want the matrix to have two rows, how many columns will there be?

#There will be three columns.

Create a matrix called mat.v using the following command:

```
mat.v <- matrix(data = v1, nrow = 2)
```

Take a look at the content of this matrix. How many columns are there? #It will have 3 columns Notice how the numbers in vec.v are used to fill up the cells of `mat.v` .We can see that R did it "by column". That is, R first filled up the first column of mat.v with the first two elements of vec.v, then moved on to the second and third columns.

You can use the `byrow` argument to change this. This argument takes one of two values, TRUE or FALSE (or T or F). That is, we write `matrix(data = v1, nrow = 2, byrow = TRUE)` Now, create an object called mat.w using the command above.

```
mat.w <- matrix(data = v1, nrow = 2, byrow = TRUE)
mat.w
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    7    9   11
```

Compare mat.v and mat.w. Do you see that R filled up the cells "by row" to create the mat.w matrix ?

Many functions in R have arguments that take TRUE or FALSE like the byrow argument we just used. In most cases, functions have a default value. In the case of the matrix function, the default value for the byrow argument is FALSE, meaning that, if you don't specify anything, R will automatically sets byrow = FALSE.

Find the number in the second row, second column of `mat.w`

```
mat.w[2, 2]
```

```
[1] 9
```

Find the number in the second row, second column of `mat.v`

```
mat.v[2, 2]
```

```
[1] 7
```

3. Lists

Create a list of months (as the names of the elements) with how many days each month has as the elements in the list

```
months_list <- list(
January = 31,
Febuary = 28,
March = 31,
April = 30,
May = 31,
June = 30,
July = 31,
August = 31,
September = 30,
October = 31,
November = 30,
December = 31
)
```

Display the number of days August has from the list

```
months_list$August
```

```
[1] 31
```

Convert the list to a vector

```
unlist(months_list)
```

```
  January   Febuary     March     April       May      June      July    August 
       31        28        31        30        31        30        31        31 
September   October  November  December 
       30        31        30        31 
```

4. Apply functions

Load R default data set mtcars

```
data(mtcars)
```

Use one of the apply functions to calculate the min value for each column/variable

```
apply(mtcars, 2, min)
```

```
   mpg    cyl   disp     hp   drat     wt   qsec     vs     am   gear   carb 
10.400  4.000 71.100 52.000  2.760  1.513 14.500  0.000  0.000  3.000  1.000 
```

Use one of the apply functions to indicate zero values in each column/variable

```
apply(mtcars, 2, function(x) x == 0)
```

```
                    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear
Mazda RX4           FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
Mazda RX4 Wag       FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
Datsun 710          FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
Hornet 4 Drive      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
Hornet Sportabout   FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Valiant             FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
Duster 360          FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Merc 240D           FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
Merc 230            FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
Merc 280            FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
Merc 280C           FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
Merc 450SE          FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Merc 450SL          FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Merc 450SLC         FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Cadillac Fleetwood  FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Lincoln Continental FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Chrysler Imperial   FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Fiat 128            FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
Honda Civic         FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
Toyota Corolla      FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
Toyota Corona       FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
Dodge Challenger    FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
AMC Javelin         FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Camaro Z28          FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Pontiac Firebird    FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE
Fiat X1-9           FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
Porsche 914-2       FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
Lotus Europa        FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
Ford Pantera L      FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
Ferrari Dino        FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
Maserati Bora       FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
Volvo 142E          FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
                     carb
Mazda RX4           FALSE
Mazda RX4 Wag       FALSE
Datsun 710          FALSE
Hornet 4 Drive      FALSE
Hornet Sportabout   FALSE
Valiant             FALSE
Duster 360          FALSE
Merc 240D           FALSE
Merc 230            FALSE
Merc 280            FALSE
Merc 280C           FALSE
Merc 450SE          FALSE
Merc 450SL          FALSE
Merc 450SLC         FALSE
```

```
Cadillac Fleetwood  FALSE
Lincoln Continental FALSE
Chrysler Imperial   FALSE
Fiat 128            FALSE
Honda Civic         FALSE
Toyota Corolla      FALSE
Toyota Corona       FALSE
Dodge Challenger    FALSE
AMC Javelin         FALSE
Camaro Z28          FALSE
Pontiac Firebird    FALSE
Fiat X1-9           FALSE
Porsche 914-2       FALSE
Lotus Europa        FALSE
Ford Pantera L      FALSE
Ferrari Dino        FALSE
Maserati Bora       FALSE
Volvo 142E          FALSE
```

Finally, execute the entire contents of this file, making sure there is no error messages.